IPE
DEC 10 2003
PATENT & TRADEMARK OFFICE

500.43289X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): S. HORIKAWA

Serial No.: 10/714,597

Filed: November 18, 2003

Title: JOB SCHEDULING MANAGEMENT METHOD, SYSTEM, AND PROGRAM

LETTER CLAIMING RIGHT OF PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

December 10, 2003

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on:

Japanese Patent Application No. 2003-193259
Filed: July 8, 2003

A certified copy of said Japanese Patent Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No.: 29,621

CIB/rr
Attachment

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００３年　７月　８日
出願番号 Application Number: [ST. 10/C]:	特願２００３－１９３２５９ [ＪＰ２００３－１９３２５９]
出願人 Applicant(s):	株式会社日立製作所

２００３年１１月１１日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００３－３０９２８４３

【書類名】　特許願
【整理番号】　K03004391A
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
【発明者】
　【住所又は居所】　神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内
　【氏名】　堀川　茂
【特許出願人】
　【識別番号】　000005108
　【氏名又は名称】　株式会社日立製作所
【代理人】
　【識別番号】　100075096
　【弁理士】
　【氏名又は名称】　作田　康夫
【手数料の表示】
　【予納台帳番号】　013088
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ジョブスケジューリング管理方法及びシステム並びにプログラム

【特許請求の範囲】

【請求項１】

ネットワークを介して接続される計算機に割り当てるジョブのスケジュールを管理するジョブスケジュール管理方法であって、前記ジョブが割り当てられた計算機の稼動状況を監視し、前記稼動状況が所定の条件を満たすか否か判断し、稼動状況が所定の条件を満たす場合、前記計算機に割り当てられたジョブのうち、前記所定の条件を満たした時点で未完了のジョブを検出し、当該検出した未完了のジョブの実行に必要なリソース情報に基づいて、当該検出した未完了のジョブを実行可能な他の計算機を抽出し、当該検出した未完了のジョブを前記抽出した他の計算機に割り当てることを特徴とするジョブスケジュール管理方法。

【請求項２】

前記所定の条件を満たすか否かの判断は、ＣＰＵ使用率が所定の使用率を超えた回数により行うことを特徴とする請求項１記載のジョブスケジュール管理方法。

【請求項３】

ネットワークを介して接続される複数の計算機にジョブを割り当て、当該ジョブのスケジュールを管理する管理計算機におけるジョブスケジュール管理方法であって、前記ジョブと当該ジョブを割り当てた計算機との対応付けを示す第１の情報と、前記ジョブの実行に必要なリソースを示す第２の情報と、各計算機で使用可能なリソースを示す第３の情報とを管理し、前記ジョブを割り当てた計算機の稼動状況を監視し、前記稼動状況が所定の条件を満たすか否か判断し、当該判断結果に応じて、前記計算機に割り当てたジョブのうち未完了のジョブを前記第１の情報を用いて検出し、当該検出した未完了のジョブの実行に必要なリソースを前記第２の情報を用いて抽出し、当該抽出されたリソースを使用可能な他の計算機を前記第３の情報を用いて抽出し、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てることを特徴とするジョブスケジュール管理方法。

【請求項４】

前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際に、前記他の計算機に既に割り当てているジョブを含めて再スケジューリングを行うことを特徴とする請求項３記載のジョブスケジュール管理方法。

【請求項５】

前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際に、前記抽出した他の計算機に既に割り当てているジョブのうち未完了のジョブを前記第１の情報を用いて検出し、当該検出した未完了のジョブの実行に必要なリソースを前記第２の情報を用いて抽出し、当該抽出されたリソースを使用可能な更に他の計算機を前記第３の情報を用いて抽出し、前記検出した未完了のジョブを前記抽出した更に他の計算機に割り当てることを特徴とする請求項３記載のジョブスケジュール管理方法。

【請求項６】

前記管理計算機は、自計算機に対してジョブを割り当てることを特徴とする請求項３記載のジョブスケジュール管理方法。

【請求項７】

前記管理計算機は、前記ジョブと当該ジョブの終了が必要な時間との対応付けを示す情報と、前記ジョブの実行に有する時間を示す情報とをさらに管理し、前記ジョブが予め定められた前記ジョブの終了が必要な時間に終了しないことを、前記ジョブを実行する計算機の稼動状況と前記ジョブの実行に有する時間から予測した場合は、前記所定の条件を満たさないとして、前記計算機に割り当てたジョブのうち未完了のジョブを他の計算機に割り当てることを特徴とする請求項３記載のジョブスケジュール管理方法。

【請求項８】

前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際、前記検出した未完了のジョブを、当該ジョブの実行に必要なリソースに応じて、複数の他の計算機に割り当てることを特徴とする請求項３記載のジョブスケジュール管理方法。

【請求項９】

ネットワークを介して接続される複数の計算機にジョブを割り当て、当該ジョブのスケジュールを管理する管理計算機であって、前記複数の計算機のうち第１の計算機に第１のジョブを割り当て、第２の計算機に第２のジョブを割り当てたことを示す情報を管理する管理手段と、前記第１の計算機の稼動状況を監視する監視手段と、当該監視手段からの指示に応じて、前記管理手段で管理する情報のうち前記第１の計算機に割り当てた第１のジョブを前記第２の計算機に割り当て直し、前記第２の計算機に割り当てた第２のジョブを第３の計算機に割り当て直す再スケジュール手段とを有することを特徴とするジョブスケジュール管理計算機。

【請求項１０】

ネットワークを介して接続される複数の計算機にジョブを割り当て、当該ジョブのスケジュールを管理する管理計算機で用いられるジョブスケジュール管理プログラムであって、前記ジョブと当該ジョブを割り当てた計算機との対応付けを示す情報と、前記ジョブの実行に必要なリソースを示す情報と、各計算機で使用可能なリソースを示す情報とを管理する機能と、前記ジョブを割り当てた計算機の稼動状況を監視する機能と、前記稼動状況が所定の条件を満たすか否か判断する機能と、当該判断結果に応じて、前記計算機に割り当てたジョブのうち未完了のジョブを検出する機能と、当該検出した未完了のジョブの実行に必要なリソースを抽出する機能と、当該抽出されたリソースを使用可能な他の計算機を抽出する機能と、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる機能とを備えることを特徴とするジョブスケジュール管理プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ジョブの運用管理を行なうためのジョブ運用管理システムに関するものであり、特にジョブスケジュール管理方法及びシステム並びにプログラムに関する。

【０００２】

【従来の技術】

コンピュータが行う複数の処理のうち、関連する一連の処理を纏めて１つのジョブとみたて、複数のジョブをスケジュールに従って実行させる技術がある。このような技術において障害が発生した場合、できるだけ早急にジョブを復帰させる必要がある。このような場合、同一ジョブが実行可能な複数の計算機を予め特定しておき、ジョブを再実行させる際には、前記予め特定された計算機のうち障害回復に最適な計算機を用いる技術がある（例えば、特許文献１参照）。

【０００３】

また、複数のジョブステップからなり、ジョブステップ間で依存関係があるジョブに対して、当該複数のジョブステップ全てを実行可能な複数の計算機を予め特定しておき、複数のジョブステップを実行する際、単一の計算機により実行するのではなく、ジョブステップ実行中に後続のジョブステップを実行するに最適な計算機を前記予め特定された複数の計算機の中から選定し、実行する技術がある（例えば、特許文献２参照）。

【０００４】

また、予め各計算機と固定的に対応付けて登録された各ジョブに対して、障害発生前に実行済みでないジョブの処理終了予定時刻を算出し、この処理終了予定時刻が、各ジョブの終了すべき時刻より前であるジョブについてのみを対象として、各ジョブの優先度に従って抽出された順に実行順序を付与してスケジュールを再構成し、再構成されたスケジュールによりジョブを実行する技術がある（例えば、特許文献３参照）。

【０００５】

【特許文献１】

特開平１１－３５３２８４号公報

【特許文献２】

特開２００１－１６６９５６号公報

【特許文献３】

特開２００１－３５０６７３号公報

【０００６】

【発明が解決しようとする課題】

上記従来技術は、障害発生時や処理負荷集中時、特定のジョブを実行可能な計算機として、特定のジョブと予め固定的に対応付けて認識されている特定の計算機を用いて代替処理を行うもので、予め特定のジョブを実行可能な計算機として対応付けられている特定の計算機の中でジョブスケジュールを再構成するものであり、ジョブの性質等（例えば、ジョブに応じて異なる実行に必要なリソース）とその稼動状況に応じて、スケジューリングの際に適宜代替となる計算機を抽出・選択するものではない。そのため、スケジュールを再構築する際に対象となるジョブは、障害ジョブやそれと依存関係があるジョブのみである。

【０００７】

また、障害ジョブやそれと依存関係があるジョブのみを考慮してスケジュールを再構築しているため、代替となる計算機のジョブ割り当て状態（例えば、既に割り当てられている依存関係のないジョブ）を考慮して、スケジュールを再構成することについては開示されていない。

【０００８】

本発明は、上記課題を考慮したジョブ運用管理技術を提供することを目的とする。

【０００９】

本発明は、ジョブを実行する計算機で障害や性能の劣化が発生した場合に他の計算機や他のジョブのスケジュールを考慮したジョブのスケジュールを再構成する技術を提供することを目的とする。

【００１０】

【課題を解決するための手段】

本発明は、ネットワークを介して接続される計算機（例えば、実行サーバ）や、計算機で実行されるエージェントプログラムに割り当てるジョブのスケジュールを管理する管理計算機（管理専用サーバであっても、実行サーバと兼用であっても良い）におけるジョブスケジュール管理方法であって、前記ジョブが割り当てられた計算機や当該計算機が利用できるリソースの稼動状況を監視したり異常を検知したりし、前記稼動状況が所定の条件を満たすか否か判断し（例えば、ＣＰＵ使用率、使用メモリ量、空きディスク容量が十分か否かの判断や、障害発生

を検知したかどうかの判断等）、稼動状況が所定の条件を満たす場合、前記計算機に割り当てられたジョブのうち未完了のジョブ（ジョブが未着手のもの、ジョブが終了していないもの、ジョブの実行が失敗したもの等）を検出し、当該検出した未完了のジョブの実行に必要なリソース情報（ＣＰＵ、メモリ、ディスク、データベース等）に基づいて、当該検出した未完了のジョブを実行可能な他の計算機を抽出し、当該検出した未完了のジョブを前記抽出した他の計算機に割り当てることを特徴とする。

【００１１】

また、ジョブと当該ジョブを割り当てた計算機との対応付けを示す第１の情報と、前記ジョブの実行に必要なリソースを示す第２の情報と、各計算機で使用可能なリソースを示す第３の情報とを管理し、前記ジョブを割り当てた計算機の稼動状況を監視し、前記稼動状況が所定の条件を満たすか否か判断し、当該判断結果に応じて、前記計算機に割り当てたジョブのうち未完了のジョブを前記第１の情報を用いて検出し、当該検出した未完了のジョブの実行に必要なリソースを前記第２の情報を用いて抽出し、当該抽出されたリソースを使用可能な他の計算機を前記第３の情報を用いて抽出し、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てることを特徴とする。

【００１２】

ここで、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際に、前記他の計算機に既に割り当てているジョブを含めて再スケジューリングを行っても良い。

【００１３】

また、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際に、前記抽出した他の計算機に既に割り当てているジョブのうち未完了のジョブを前記第１の情報を用いて検出し、当該検出した未完了のジョブの実行に必要なリソースを前記第２の情報を用いて抽出し、当該抽出されたリソースを使用可能な更に他の計算機を前記第３の情報を用いて抽出し、前記検出した未完了のジョブを前記抽出した更に他の計算機に割り当てても良い。

【００１４】

また、前記管理計算機は、前記ジョブと当該ジョブの終了が必要な時間との対応付けを示す情報と、前記ジョブの実行に有する時間を示す情報とをさらに管理し、前記ジョブが予め定められた前記ジョブの終了が必要な時間に終了しないことを、前記ジョブを実行する計算機の稼動状況と前記ジョブの実行に有する時間から予測した場合は、前記所定の条件を満たさないとして、前記計算機に割り当てたジョブのうち未完了のジョブを他の計算機に割り当てても良い。

【００１５】

また、前記検出した未完了のジョブを前記抽出した他の計算機に割り当てる際、前記検出した未完了のジョブを、当該ジョブの実行に必要なリソースに応じて、複数の他の計算機に割り当てても良い。

【００１６】

尚、上記目的を達成するためには、上述した機能を実現するシステム、プログラム若しくはプログラムを格納した記録媒体であっても良い。

【００１７】

【発明の実施の形態】

以下本発明の実施の形態を、図面を参照しながら詳細に説明する。図１は、ジョブを実行するシステム構成の一実施形態を示す全体構成図である。

【００１８】

本システムは、ネットワーク１３０を介してジョブの運用管理を行う管理計算機（本実施例では、サーバ）１００、ジョブの実行をする実行計算機（本実施例では、サーバ１、サーバ２、サーバ３；１０１）が接続されている。実行サーバ１０１は１台であっても、複数であってもよいが、実行サーバで実行されるジョブに依存関係がない場合は、システム全体として複数のエージェントを有することとなる。

【００１９】

管理計算機（サーバ）１００は、専用計算機（サーバ）であっても、実行計算機（サーバ）１０１と併用であっても良い。本実施形態では、管理計算機（サーバ）１００を専用計算機（サーバ）とし、情報を表示する表示部やキーボード・マウス等の入力部等からなる入出力部を有し、ジョブ処理システムの操作を行う

コンソール１０３が接続されている。実行計算機（サーバ）１０１には、各実行サーバで利用可能なリソース１、リソース２（１０２）が接続されている。

【００２０】

ここで、リソースとは、ＣＰＵ、メモリ、データベース、ディスク装置、ストレージなどのジョブ実行に利用されるものであり、稼動性能を取得できるものであれば良く、本実施例では、リソース１、リソース２（１０２）が実行計算機（サーバ）１０１に対して外付けで表示されているが、実行計算機（サーバ）１０１内にあるものでも良い。また、リソース１、リソース２（１０２）と実行計算機１０１との間をＳＡＮ（Storage Area Network）環境で構築可能である。本発明はフェイルオーバやディザスタリカバリの際のスケジュール管理に適用可能である。

【００２１】

管理計算機（サーバ）１００と実行計算機（サーバ）１０１は、大型コンピュータやサーバシステムなどで、汎用のコンピュータであっても、専用のハードウェアにより構成されているものであってもよく、システム全体の制御及びプログラム処理を行うマイクロチップなどの処理部（制御部）（１１１、１２２）と、プログラムやデータを保持する記憶部（記憶装置）（１１２、１２３）とを含んで構成されている。

【００２２】

記憶部（１１２、１２３）はメモリ等の主記憶装置やハードディスク等の補助記憶装置、データベース等から構成され、記憶部１１２にはマネージャプログラムの他、管理サーバ１００の処理上保持すべきデータやプログラムが記憶されている。記憶部１２３にはエージェントプログラム、アプリケーションプログラムの他、実行サーバ１０１の処理上保持すべきデータやプログラムが記憶されている。

【００２３】

管理サーバ１００の機能と実行サーバ１０１の機能とは、その用途に応じて、別個独立のシステムとして管理されても、一体化されたシステムとして提供されても良い。また、記憶部１１２、１２３の機能の一部を、管理サーバ１００や実

行サーバ１０１の外部に存在する固有データベースや共通データベース等として管理しても良い。

【００２４】

記憶部（１１２、１２３）に格納されたプログラム等と処理部（１１１、１２２）とが連動して、管理サーバ１００では、ジョブ自身やジョブを管理するジョブ処理システムのマネージャ１１０、実行サーバ１０１では、ジョブ自身や、ジョブを実行するリソースや実行サーバで動作しているアプリケーション１２１の稼動性能を収集するジョブ処理システムのエージェント１２０を実行する。アプリケーション１２１は、データベースアプリケーションなどの一般のアプリケーションである。

【００２５】

各実行サーバの稼動性能や利用できるリソース、アプリケーションの種類は異なる場合がある。

【００２６】

図２は、マネージャ１１０の主要機能の一実施形態を示す機能ブロック図である。

【００２７】

マネージャ１１０は、実行サーバ１０１やエージェント１２０に割り当てるジョブのスケジュールを管理するスケジュール管理部２００、管理サーバ１００が管理している実行サーバ１０１上のエージェント１２０を管理するエージェント管理部２０１、各実行サーバから利用できるリソース１０２や各実行サーバ上のアプリケーション１２１や処理部１２２を管理するリソース管理部２０２、現在ジョブを実行しているエージェント１２０を管理するジョブ実行中エージェント管理部２０３、実行サーバ１０１で障害が発生したり、性能が劣化した場合にジョブのスケジュールの再構成を行う再スケジューリング部２０４、ジョブの実行に必要なリソースを有する実行サーバ１０１や当該実行サーバに存在するエージェントを検索する代替エージェント検索部２０５、マネージャ１１０や他のエージェント１２０と通信するための通信部２０６とを有する。

【００２８】

マネージャ１１０は、管理サーバ１００の記憶部１１２に、ジョブの実行される時間や実行される実行サーバの情報を管理しているスケジュールテーブル３００（後述）、実行サーバ１０１で実行されるジョブを管理したジョブテーブル４００（後述）、マネージャの管理しているエージェントの情報を管理したエージェント管理テーブル５００（後述）、またジョブ処理システム内で利用可能な全てのリソースとそれを使用できる実行ノードの関連を管理しているリソース間距離テーブル６００（後述）を適宜格納したり、前記テーブルに格納されるデータを参照することによって本実施形態を実現する。

【００２９】

図３は、スケジュールテーブル３００のデータ構成の一例を示す図である。

【００３０】

このスケジュールテーブル３００は、管理者によって設定され、再スケジュールの際にはマネージャ１１０によって変更される。マネージャ１１０のスケジュール管理部２００は、スケジュールテーブル３００を常に監視し，このテーブルに従って通信部２０６を介してジョブの実行命令を実行サーバ１０１に伝える。ジョブの実行命令を伝えた後，ジョブ実行中エージェント管理部２０３が実行サーバ１０１上のエージェントの監視を開始する。

【００３１】

スケジュールテーブル３００は，ジョブの運用を管理するための情報である。具体的には、実行開始日時３０１、最終終了日時３０２、最大実行時間３０３、ジョブを実行する実行エージェント３０４、実行ジョブ３０５に関する情報が含まれる。実行開始日時３０１は、ジョブを開始する日時を示し、実行を行う日、時間を含む。定期的に実行するジョブであるならば、その起動時刻を示す。最終終了日時３０２は、ジョブが最終的に終了していなければいけない時間や最終的に終了していて欲しい時間を示す。特に制限がない等の場合は、必ずしも設定される必要はない。最大実行時間３０３は、ジョブの最大実行時間や期間を示し、ジョブの実行時間がこの値を超える場合を、マネージャ１１０によるスケジュールの再構成のきっかけとすることができる。実行エージェント３０４は、ジョブを実行するために実行サーバ１０１に存在するエージェントを示す。実行ジョブ

３０５は、実行されるジョブを示す。

【００３２】

図４は、ジョブテーブル４００のデータ構成の一例を示す図である。

【００３３】

ジョブテーブル４００は、管理者が設定するテーブルで、ジョブの実行内容を管理するテーブルであり、実行に必要な情報を含む。マネージャ１１０のスケジュール管理部２００は、スケジュールテーブル３００から実行するジョブを選択し、そのジョブの内容をこのテーブルから取得する。

【００３４】

具体的には、ジョブ名４０１、実行内容４０２、実行必要リソース４０３、最大許容コスト４０４、優先度４０５に関する情報を含む。ジョブ名４０１は、ジョブの名称を示す。実行内容４０２は、そのジョブが実行する処理内容を示す。実行必要リソース４０３は、そのジョブを実行するために必要なリソース（ＣＰＵ、メモリ、ディスク、データベースソフト、その他のリソース１、その他のリソース２等）を表す。例えば、データベースアプリケーションソフトとディスク装置が必要なジョブならば、それらの必要なリソースの条件が設定される。最大許容コスト４０４は、ジョブを１度実行する時に消費して良い最大のコストを示す。

【００３５】

ここで、コストとは、リソースを使用する場合の効率を表す値である。この値が小さいほどジョブの実行を効率良く行う事ができることを示す。コストの値は、例えば、ジョブを実行するために必要なリソースの使用単位コストの和から算出する。優先度４０５は、各ジョブの優先度を示す。本実施形態では、優先度の高いものをA、低いものをＣとしている。

【００３６】

図５は、エージェント管理テーブル５００のデータ構成の一例を示す図である。

【００３７】

エージェント管理テーブル５００は、マネージャ１１０が管理するエージェン

ト１２０に関する情報を管理するテーブルであり、エージェントが実行されている実行サーバ１０１との関連を示す。

【００３８】

具体的には、エージェント名５０１は、エージェントの名称を示す。マネージャ１１０のスケジュール管理部２００は、スケジュールテーブル３００から実行するエージェント１２０を取得し、そのエージェント１２０が動作している実行サーバ１０１を取得する。例えば、エージェント２とエージェント４とが実行サーバ２に対応付けられているように、１台の実行サーバ１０１に複数のエージェント１２０を設定することができる。複数のエージェントが存在する実行サーバは、複数のジョブ（特に依存関係のない複数のジョブ）を並行して処理したりする場合にも対応可能である。

【００３９】

図６は、リソース間距離テーブル６００のデータ構成の一例を示す図である。

【００４０】

リソース間距離とは、実行サーバ１０１からリソースを使用するときにかかる単位コストを示す。この値は、マネージャ１１０がエージェントから取得した稼働性能を元にリソース管理部２０２を介して随時変更される。本実施形態では、リソースが使用できない場合（リソースが割り当てられていない場合、障害等で一時的に使用できない場合など）、単位コストは表示されていない。

【００４１】

リソースが使用できない場合に、そもそもリソースが割り当てられていないのか、障害等で一時的に使用できないのか（どの程度で回復見込みか、使用可能となった場合の単位コストはどの程度か）といった状態が判断可能な情報も保持することによって、スケジュール再構築の際の一判断材料とすることも可能である。

【００４２】

具体的には、使用可能リソース６０１は、マネージャ１１０が管理するエージェント１２０が使用可能な全てのリソースを示す。使用実行サーバ６０２は、使用可能リソース６０１を使用できる実行サーバ１０１を示す。例えば、実行サー

バ１のＣＰＵを使用できる実行サーバは、実行サーバ１でその使用単位コストは５であることを示し（６０３）、逆に実行サーバ２、実行サーバ３では、使用できないことを示す。また、リソース１のディスクは，実行サーバ１と実行サーバ３で使用可能であることを示す（６０４、６０５）。

【００４３】

またリソースの使用単位コストは、各実行サーバ１０１上のエージェント１２０が収集したリソースの稼動性能情報により算出される。例えば、ディスクの単位コストは，実行サーバ１０１から使用した場合の平均使用時間やＩ／Ｏオペレーションの平均実行時間などから算出する。そのため同じリソースを使用した場合でも、使用する実行サーバ１０１により異なる。単位コストの値が小さいほど実行サーバ１０１からのリソースの性能が良い事を示す。

【００４４】

尚、上述した実施形態ではスケジュールテーブル３００とジョブテーブル４００とエージェント管理テーブル５００とリソース間距離テーブル６００とを分けて管理しているが、適宜、一連のデータとして管理しても、データ項目の組み合わせを変更しても良い。

【００４５】

図７は、マネージャ１１０の基本シーケンスの一例を説明するフローチャートである。

【００４６】

マネージャ１１０のエージェント管理部２０１は、新規のエージェント１２０からの新規登録要求を受信した場合（Ｓ７００）、その新規エージェント１２０を実行サーバと対応付けてエージェント管理テーブル５００に登録する（Ｓ７０１）。マネージャ１１０のジョブ実行中エージェント管理部２０３は、実行サーバやリソースの稼動状況を監視する手法として、例えば、定期的にエージェントへの問合せを行ったり、障害発生に応じてエージェントから報告を受けたりする。

【００４７】

定期的に各エージェント１２０から稼動性能情報を収集する場合は、パフォーマンス収集時間がきたかどうか判断し（Ｓ７０２）、パフォーマンス収集時間がき

た場合は、管理する全てのエージェント１２０に現在の稼動性能情報を送信するよう要求する（Ｓ７０３）。送信された稼動性能情報を元にリソース管理部２０２を介してリソース間距離テーブルを更新する（Ｓ７０４）。

【００４８】

マネージャ１１０のスケジュール管理部２００は、スケジュールテーブル３００を検索し、実行時間になったジョブがあった場合（Ｓ７０５）、ジョブの実行命令を送信する（Ｓ７０６）。ジョブ実行命令を送信すると、マネージャ１１０のジョブ実行中エージェント管理部２０３は、ジョブの実行状態とそれを実行しているエージェント１２０の状態を監視することとなる。

【００４９】

図８は、エージェント１２０の主要機能の一実施形態を示す機能ブロック図である。

【００５０】

エージェント１２０は、マネージャ１１０から割り当てられたジョブの実行処理を行うジョブ実行部８００、稼動性能情報の収集を行うパフォーマンス収集部８０１、収集した稼動情報から現在の実行サーバ１０１の状態を分析し、ジョブの実行に最適なリソースの使用状態を計算するリソース使用状況管理部８０２、マネージャ１１０や他のエージェント１２０と通信を行う通信部８０３とを有する。

【００５１】

エージェント１２０は、実行サーバ１０１の記憶部１２３に、収集した稼動性能情報を蓄積するための履歴パフォーマンステーブル９００（後述）、リソース異常状態を示す条件を保存するリソース使用制限テーブル１０００（後述）を適宜格納したり、前記テーブルに格納されるデータを参照することによって本実施形態を実現する。

【００５２】

エージェント１２０は，マネージャ１１０からジョブの実行命令を受け付けるとジョブ実行部８００でジョブを実行する。

【００５３】

図９は、履歴パフォーマンステーブル９００のデータ構成の一例を示す図である。

【００５４】

履歴パフォーマンステーブル９００は、エージェント１２０のパフォーマンス収集部８０１が定期的・不定期に収集した稼動性能を蓄積する度に更新する。具体的には、収集時刻９０１、収集項目９０２から構成される。収集時刻９０１は、エージェント１２０が稼動性能を収集した時刻が格納される。収集項目９０２は、収集した稼動性能を項目毎に格納する。例えば、ＣＰＵ使用率９０３の稼動性能が格納される。他にも、使用メモリ量や空きディスク容量、ディスク(平均ディスク処理時間９０４)やデータベースアプリケーション(平均クエリ処理時間９０５)などの稼動性能が格納される。

【００５５】

図１０は、リソース使用制限テーブル１０００のデータ構成の一例を示す図である。

【００５６】

リソース使用制限テーブル８０５は、管理者が設定、もしくはエージェント１２０のリソース使用状況管理部８０２が更新するテーブルであり、エージェント１２０のパフォーマンス収集部８０１が収集した稼動性能から現在のリソースの状態を判定する条件を示す。エージェント１２０のパフォーマンス収集部８０１がジョブ実行中に稼動性能を収集する度に判定される。

【００５７】

具体的には、ジョブ１００１、しきい値１００２、しきい値超え回数１００３を含んで構成される。ジョブ１００１は、適用されるジョブの名称を示す。しきい値１００２は、収集した稼動性能からリソースの異常状態かどうかを判断する条件を示す。しきい値１００２に指定される条件は、収集する稼動性能の一部を指定する。例えば、ジョブ１は、空きディスク容量が２００ＭＢ以下である場合にしきい値超えとして認識され、カウントされる。しきい値は１つのジョブ１００１に対して複数指定できる。しきい値超え回数１００３は、しきい値１００２を超えた回数を指定する。例えば、ジョブ１は、稼動性能を３回収集したうちの

２回，しきい値１００２の条件を超えた場合に、異常状態として認識される。

【００５８】

図１１は、エージェント１２０の基本シーケンスの一例について説明するフローチャートである。

【００５９】

エージェント１２０のパフォーマンス収集部８０１は、稼動性能収集時間がきたかどうか判断し（Ｓ１１００）、定期的に実行サーバ１０１とそれから使用できるリソース１０２の稼動性能を収集し(Ｓ１１０１)、履歴パフォーマンステーブル９００を更新する。マネージャ１１０からのジョブ実行命令を受信した場合（Ｓ１１０２）、ジョブ実行部８００を介してジョブを実行する（Ｓ１１０３）。ジョブの実行中は，常に稼動性能を収集し（Ｓ１１０４）、リソース使用状況管理部８０２を介してリソース使用制限テーブル１０００の条件を照らし合わせ、実行サーバ１０１もしくはリソース１２０の状態を監視し（Ｓ１１０５）、リソースの状態が異常状態になった場合は、通信部８０３を介してマネージャに異常が発生した事を通知する（Ｓ１１０６）。ステップ１１０４からステップ１１０６の処理をジョブが終了するまで繰り返し、ジョブが終了した場合（Ｓ１１０７）は、ジョブの使用したリソースとその状態やジョブの最大実行時間３０３とジョブの実際の実行時間から、リソース使用状況管理部８０２を介して次回のリソース使用制限を設定する（Ｓ１１０８）。

【００６０】

図１２は、リソース使用制限の設定（Ｓ１１０８）の一例を示すフローチャートである。リソース使用制限テーブル１０００に実行したジョブの条件が設定されていない場合（Ｓ１２００）と、ジョブが最大終了時間３０３で終了した場合（Ｓ１２０１）は、最新のジョブが使用したリソースの稼動性能を条件として設定する。それ以外は、リソース使用制限テーブル１０００の更新は行わない。

【００６１】

図１３は、マネージャ１１０における、実行中のジョブとそれを実行しているエージェント１２０の監視処理のフローチャートである。

【００６２】

マネージャ１１０のジョブ実行中エージェント管理部２０３は、ジョブが終了するまでエージェント１２０を監視する（Ｓ１３００～Ｓ１３０６）。

【００６３】

ジョブの実行命令を送信した時やジョブの実行中にエージェントからの応答がない場合等、エージェントで異常が発生した場合（Ｓ１３００）、エージェント１２０が使用しているリソースが異常状態(リソースの使用制限の超過やリソース自体の故障)になった場合（Ｓ１３０１）は、実行中のジョブの実行が失敗した（未完了である）ものと判断し、再スケジューリング部２０４にて実行中のジョブの再スケジューリングを行い，ジョブの再実行を行う（Ｓ１３０２）。ジョブの最大実行時間を経過してもジョブが終了していない場合（Ｓ１３０３）は、実行中のジョブの次に予定されているジョブをスケジュール管理部２００を介してスケジュールテーブル３００から検索し（Ｓ１３０４）、そのジョブに対してスケジュールの再設定を行う（Ｓ１３０５）。ジョブが終了し（Ｓ１３０６）、ジョブの実行時間が最終終了日時３０３を超過した場合（Ｓ１３０７）は、ワーニングメッセージをコンソール１０３に出力し，管理者に伝える（Ｓ１３０８）。

【００６４】

図１４は、マネージャ１１０の再スケジューリング部２０４における処理を示すフローチャートである。

【００６５】

再スケジューリングを行う際、マネージャ１１０の代替エージェント検索部２０５にて同じジョブを実行できるエージェント１２０の検索を、ジョブテーブル４００やエージェント管理テーブル５００やリソース間距離テーブル６００を用いて、エージェントが存在する実行サーバから使用可能なリソースに基づいて行う（Ｓ１４００）。随時リソースの稼動性能情報が反映されるリソース間距離テーブル６００を用いて、再スケジュールの際に代替エージェントを検索するので、リソースの稼動状況に応じたジョブの割り当てが可能となる。

【００６６】

ここで代替となるエージェント１２０が１つ以上見つかった場合（Ｓ１４０１

）、そのエージェント１２０の中にジョブの実行する時間帯のスケジュールが空いているエージェントをスケジュールテーブル３００を用いて検索する（Ｓ１４０２）。スケジュールの空いたエージェントがあった場合（Ｓ１４０２で「Yes」）、スケジュールテーブル３００の実行エージェント３０４を変更する（Ｓ１４１１）。

【００６７】

スケジュールの空いたエージェント１２０が存在しない場合（Ｓ１４０２で「No」）は、ステップ１４０１で検索した結果のエージェントに対して同時間帯のスケジュールの再設定を行う。再設定では、同時間帯にスケジュールされたジョブを、ジョブテーブル４００の優先度４０５に基づいて、優先度順に並び替える（Ｓ１４０３）。次に、遅延が発生するジョブより優先度が低い、若しくは同じジョブがある場合（Ｓ１４０４で「Yes」）は、低優先度に対するジョブの再スケジューリングを行う（Ｓ１４０５）。ここで遅延が発生しないジョブの再スケジューリングが行えた場合（Ｓ１４０６で「Yes」）は、スケジュールテーブル３００を変更し、ジョブの実行エージェントを変更する(Ｓ１４１１)。

【００６８】

再スケジューリングにより遅延が発生する、若しくはジョブの再スケジューリングを行う事ができない場合は、遅延が発生するジョブより高い優先度をもつジョブに対して，再スケジューリングの検索を行う。最初に遅延が発生するジョブより高い優先度をもつジョブが存在するかを確認し、存在する場合（Ｓ１４０７で「Yes」）は、検索処理を行う（Ｓ１４０８）。検索処理の結果、遅延が発生しないスケジュールが検索できた場合（Ｓ１４０９で「Yes」）ジョブを置き換えるため、スケジュールテーブル３００を変更し、ジョブの実行エージェントを変更する（Ｓ１４１１）。遅延が発生しないスケジュールが検索できない場合（Ｓ１４０９で「No」）は、低優先度検索処理１４０５の結果から遅延の最小となるジョブの置き換えを採用し、スケジュールテーブル３００を再設定し（Ｓ１４１０）、コンソール１０３にワーニングメッセージを出力する（Ｓ１４１２）。

【００６９】

また代替となるエージェント１２０が１つもない場合は，エラーメッセージを

コンソール１０３に出力する（Ｓ１４１３）。

【００７０】

再スケジューリング部の処理は、ステップ１４００やステップ１４０１で検索した結果のエージェントに対して再スケジューリング部の処理を行うため、最初の呼び出し以外は再スケジューリング部内部から再起的に呼び出されるようにすることによって、再スケジュールに関連するエージェントに対し、順次再スケジュールを行うことが可能となる。

【００７１】

尚、本フローは適宜順序を改変可能であり、実施形態に応じては、ジョブに優先度を設けず、優先度検索処理を省略することも可能である。

【００７２】

図１５は低優先度検索処理１４０５の一実施形態を示すフローチャートである。

【００７３】

低優先度検索処理では、置き換えの対象となるジョブをＪｏｂ１とし（Ｓ１５０１）、Ｊｏｂ１と優先度が同じ、若しくは低いジョブそれぞれに対して検索処理を行う（Ｓ１５０２～Ｓ１５１２）。最初に、優先度の同じ、若しくは低いジョブの１つをＪｏｂ２に設定する（Ｓ１５０３）。Ｊｏｂ１とＪｏｂ２を置き換えた場合に、Ｊｏｂ２の開始時間に遅れる事なく再スケジュールができる場合は（Ｓ１５０５で「Yes」）、この組み合わせを採用し，検索処理を終了する（Ｓ１５１３）。

【００７４】

遅延が発生する場合（Ｓ１５０５で「No」）は、遅延時間を計算する（Ｓ１５０６）。遅延時間の計算は、置き換え後の実行開始日時３０１を求め、その時点から最大実行時間３０３を足すことで、最終終了日時３０２からどのくらい遅延するかで行う。

【００７５】

次に、Ｊｏｂ２に対して、再度低優先度検索処理（高優先度検索処理）を呼び出す（Ｓ１５０７）。これにより再度ステップ１５００からの処理を再帰的に呼

び出す事で置き換えの対象となるジョブの組み合わせを検索する。

【００７６】

ステップ１５０７の結果、Ｊｏｂ２が遅延なしで再スケジューリングができる場合（Ｓ１５０８で「Yes」）は検索を終了し、検索結果を再スケジューリングの結果として採用する（Ｓ１５１３）。

【００７７】

遅延なしの組み合わせが検索できない場合（Ｓ１５０８で「No」）は、先ほど計算した遅延時間がこれまでの検索結果より小さいかを判定し（Ｓ１５０９）、小さい場合はジョブ置き換えの組み合わせとその遅延時間を保存する（Ｓ１５１１、Ｓ１５１２）。

【００７８】

図１６は高優先度検索処理１４０８の一実施形態を示すフローチャートである。

【００７９】

高優先度検索処理では、置き換えの対象となるジョブをＪｏｂ１とし（Ｓ１６０１）、Ｊｏｂ１と優先度が高いジョブそれぞれに対して検索処理を行う（Ｓ１６０２～Ｓ１６０９）。最初に、優先度の高いジョブの１つをＪｏｂ２に設定する（Ｓ１６０３）。Ｊｏｂ１とＪｏｂ２を置き換えた場合に、Ｊｏｂ２の開始時間に遅れる事なく再スケジュールができる場合は（Ｓ１６０５で「Yes」）、この組み合わせを採用し，検索処理を終了する（Ｓ１６０８）。

【００８０】

遅延が発生する場合（Ｓ１６０５で「No」）は、Ｊｏｂ２に対して、再度高優先度検索処理（低優先度検索処理）を呼び出す（Ｓ１６０６）。これにより再度ステップ１６００からの処理を再帰的に呼び出す事で置き換えの対象となるジョブの組み合わせを検索する。

【００８１】

ステップ１６０６の結果、Ｊｏｂ２が遅延なしで再スケジューリングができる場合（Ｓ１６０７で「Yes」）は検索を終了し、検索結果を再スケジューリングの結果として採用する（Ｓ１６０８）。遅延なしの組み合わせが検索できない場

合（Ｓ１６０７で「No」）は、採用せずに終了する。

【００８２】

図１７は、マネージャ１１０における代替エージェント検索部２０５の処理（図１４のステップ１４００における処理）の一例を示すフローチャートである。

【００８３】

スケジュールの置き換えを行うジョブの利用するリソースをジョブテーブル４００から取得する（Ｓ１７００）。次にリソース間距離テーブル６００から必要な全てのリソースを使用できる実行サーバ１０１を検索する（Ｓ１７０１）。ここでリソースを使用できる実行サーバ１０１が存在した場合（Ｓ１７０２）、実行コストを計算する（Ｓ１７０３）。実行コストは，ジョブテーブル４００内の最大許容コスト以下になるようにし（Ｓ１７０４）、そのようなリソースの組み合わせがある実行サーバ１０１とそこで動作するエージェント１２０を選択する（Ｓ１７０５）。

【００８４】

尚、ステップ１７０１でリソース間距離テーブルから必要な全てのリソースを使用できる実行サーバを検索する際、複数の実行サーバで使用可能なリソースを組み合わせることにより、複数の実行サーバとそこで動作するエージェントを特定しても良い。

【００８５】

図１８はジョブのスケジュール再設定の具体例を示す。

【００８６】

図１８（ａ）のようにジョブがスケジュールされていた場合を考える。またこのときのジョブテーブル４００とリソース間距離テーブル６００はそれぞれ図４と図６であるとする。

【００８７】

ジョブ１が終了予定時刻より遅延する（１８００）。マネージャ１１０のジョブ実行中エージェント管理部２０３は、ジョブ１の遅延を検知し、他のジョブに影響があるかを検索し、次に実行される予定であるジョブ２へ影響がある事を検知する（１８０１）。

【００８８】

ここでマネージャ１１０は、図１４で述べた方法を用いて、ジョブ２の再スケジューリングを行う。ジョブ２を実行するためには、ＣＰＵとメモリとディスクを必要とする（図４）。ジョブ２を実行サーバ２と実行サーバ３で実行可能であるが（図６）、最大許容コストを計算すると実行サーバ２は１５、実行サーバ３は２５であるため実行サーバ２のみ代替可能である（図４と図６）。そこで、ジョブ２を実行サーバ２で実行するように再スケジューリングする（１８０２、１８０３）。しかし同時刻に実行サーバ２では，ジョブ３がスケジュールされている（１８０４）。そこでジョブ３のスケジュールの再構成を行う。ジョブ３を実行できる実行サーバは、実行サーバ３のみ（図６）であるため，ジョブ３を実行サーバ３で実行するようにする（１８０５、１８０６）。しかしジョブ４が同時刻にスケジュールされているため（１８０７）、ジョブ４の再スケジューリングが必要となる。ジョブ４を実行できる実行サーバは実行サーバ１のみ（図６）である。実行サーバ１は、本来ならばジョブ２の実行がスケジュールされていたが、これまでのスケジュールの再構成により，スケジュールの空きができている（１８０２）。これによりジョブ４の再スケジュールが終了する（１８０７、１８０８）。

【００８９】

以上より全てのジョブに対してスケジュールの再構成を行う事ができ，ジョブ１の遅延による影響を最小限に留める事ができる。

【００９０】

図１９は、優先度を加味した場合のジョブのスケジュール再設定の具体例を示す。

【００９１】

図１９(a)のようにジョブがスケジュールされていた場合を考える。またこのときのジョブ２は、実行サーバ１と実行サーバ２と実行サーバ３のいずれでも実行可能とし、ジョブ３は、実行サーバ１と実行サーバ２で実行可能、ジョブ４は、実行サーバ１と実行サーバ３で実行可能であるとする。ジョブの優先度は、それぞれジョブ２が優先度Ｂ、ジョブ３が優先度Ａ、ジョブ４が優先度Ｃとし、ジ

ョブ３が最も優先度が高いものとする。

【００９２】

ジョブ１が終了予定時刻より遅延する（１９００）。マネージャ１１０は、ジョブ１の遅延を検知し、他のジョブに影響があるかを検索し，次に実行される予定であるジョブ２へ影響がある事を検知する（１９０１）。

【００９３】

ここでマネージャ１１０は，図１４で述べた方法を用いて，ジョブ２の再スケジューリングを行う。再スケジューリングを行う実行サーバとしては、実行サーバ２、実行サーバ３の二つがあるが、それぞれ同時間帯にジョブ３とジョブ４がスケジュールされている（１９０２、１９０３）。ここでジョブ３と置き換えを考えた場合、ジョブ３が実行サーバ１に移動するが，遅延が発生する（１９０４）。しかしジョブ３はジョブ２より優先度が高いため、この置き換えを行う事ができない（１９０５）。

【００９４】

ジョブ４との置き換えを行う場合（１９０６）、ジョブ３同様にジョブ４も遅延するが、ジョブ４の優先度がジョブ２より低いため遅延する場合でも、再スケジューリングを行う（１９０７）。

【００９５】

またこの際コンソール１０３にワーニングメッセージを出力し，遅延が発生する事を管理者に知らせる（１９０８）。

【００９６】

以上説明したように、本実施形態によれば、ジョブを実行する計算機で障害や性能の劣化が発生した場合に他の計算機や他のジョブのスケジュールを考慮したジョブのスケジュールを再構成する技術を提供可能となる。また、利用可能なリソースの追加や削除に柔軟に対応でき、管理者がジョブ実行のスケジュールの設定を変更せずとも，常にリソースを有効に利用したジョブ処理と運用管理を行うことが可能となる。

【００９７】

上述した実施形態は、本発明の主旨を逸脱しない範囲で適宜変更や組み合わせ

可能である。

【００９８】

【発明の効果】

本発明によれば、リソースの稼動状況に応じたジョブスケジュールを適宜柔軟に提供可能となる。

【図面の簡単な説明】

【図１】

ジョブを実行するシステム構成の一実施形態を示す全体構成図

【図２】

マネージャの主要機能の一実施形態を示す機能ブロック図

【図３】

スケジュールテーブルのデータ構成の一例を示す図

【図４】

ジョブテーブルのデータ構成の一例を示す図

【図５】

エージェント管理テーブルのデータ構成の一例を示す図

【図６】

リソース間距離テーブルのデータ構成の一例を示す図

【図７】

マネージャの基本処理の一例を示すフローチャート

【図８】

エージェントの主要機能の一実施形態を示す機能ブロック図

【図９】

履歴パフォーマンステーブルのデータ構成の一例を示す図

【図１０】

リソース使用制限テーブルのデータ構成の一例を示す図

【図１１】

エージェントの基本処理の一例を示すフローチャート

【図１２】

リソース使用制限テーブルの更新処理の一例を示すフローチャート

【図１３】

マネージャのジョブ実行中エージェント管理部における処理の一例を示すフローチャート

【図１４】

マネージャの再スケジューリング部における処理の一例を示すフローチャート

【図１５】

低優先度検索処理の一実施形態を示すフローチャート

【図１６】

高優先度検索処理の一実施形態を示すフローチャート

【図１７】

マネージャの代替エージェント検索部の処理の一例を示すフローチャート

【図１８】

ジョブ実行のスケジュールの再構成の一実施形態

【図１９】

ジョブ実行のスケジュールの再構成の他の実施形態

【符号の説明】

１００ 管理サーバ

１０１ 実行サーバ

１０２ リソース

１０３ コンソール

１１０ マネージャ

１１１ 処理部

１１２ 記憶装置（記憶部）

１２０ エージェント

１２１ アプリケーション

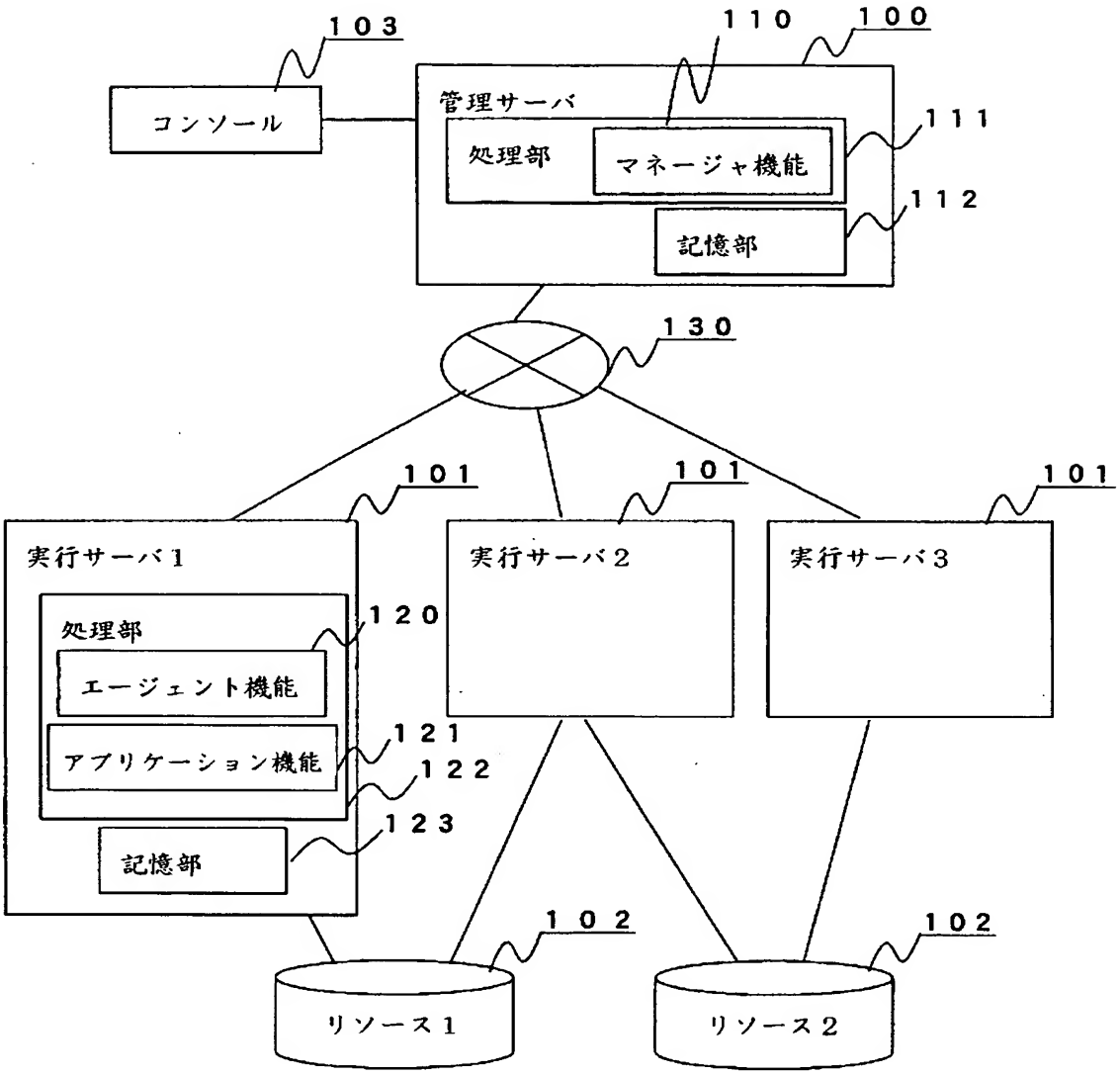
１２２ 処理部

１２３ 記憶装置（記憶部）

【書類名】　図面

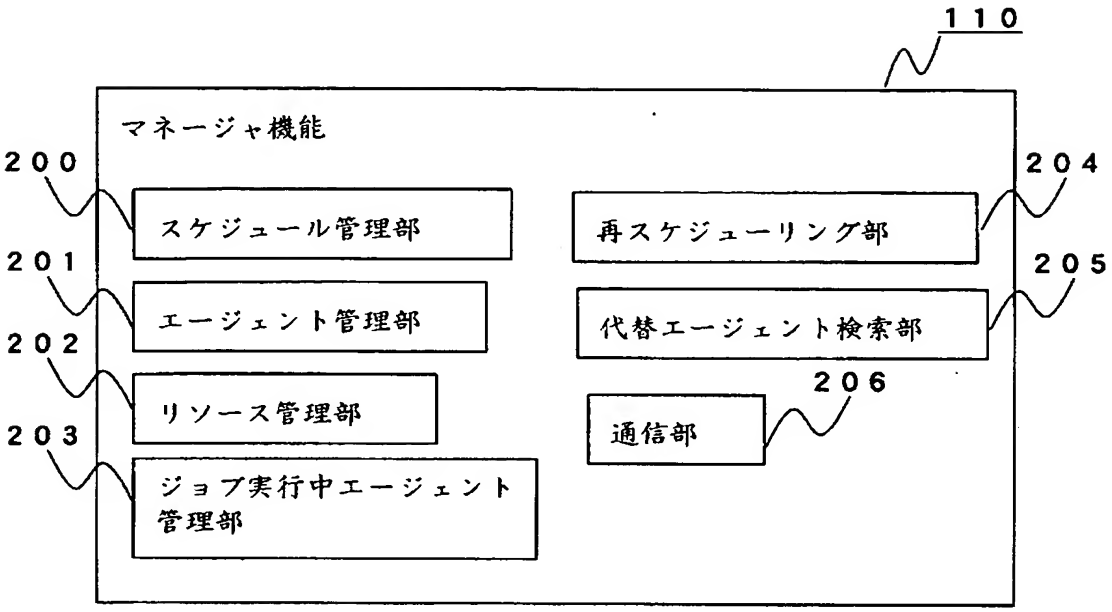
【図１】

図１


１０３
１１０
１００
コンソール
管理サーバ
処理部
マネージャ機能
１１１
１１２
記憶部
１３０
１０１
１０１
１０１
実行サーバ１
実行サーバ２
実行サーバ３
処理部
１２０
エージェント機能
アプリケーション機能
１２１
１２２
１２３
記憶部
１０２
１０２
リソース１
リソース２

【図２】

図２



【図３】

図３

スケジュールテーブル　３００

３０１ 実行開始日時	３０２ 最終終了日時	３０３ 最大実行時間	３０４ 実行エージェント	３０５ 実行ジョブ
2002/11/10 10:00:00	2002/11/10 11:00:00	01:00	エージェント１	ジョブ１
2002/11/10 23:00:00	2002/11/11 00:00:00	00:30	エージェント３	ジョブ２
毎週水曜 12:00:00	-	02:00	エージェント２	ジョブ３
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

【図４】

図４

ジョブテーブル　４００

４０１ ジョブ名	４０２ 実行内容	４０３ 実行必要リソース	４０４ 最大許容コスト	４０５ 優先度
ジョブ１	計算処理	CPU，メモリ	10	A
ジョブ２	バッチ処理	CPU，メモリ，ディスク	20	B
ジョブ３	DB処理	CPU，メモリ，データベースソフト，リソース２	70	A
ジョブ４	DB処理	CPU，メモリ，データベースソフト，リソース１	30	C
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

【図５】

図５

エージェント管理テーブル　５００

501 エージェント名	502 実行サーバ
エージェント１	実行サーバ１
エージェント２	実行サーバ２
エージェント３	実行サーバ３
エージェント４	実行サーバ２
・ ・ ・	・ ・ ・

【図６】

図６

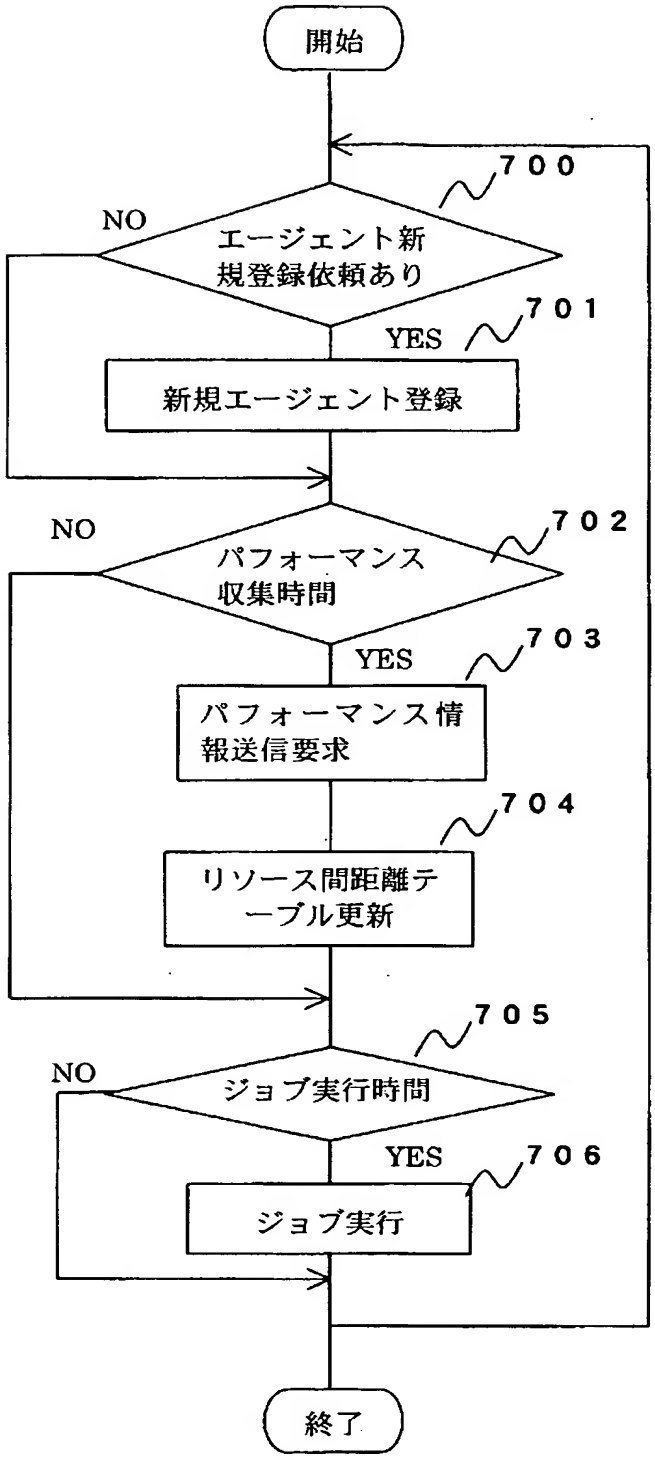
リソース間距離管理テーブル　６００

使用可能リソース (601)			使用実行サーバ (602)			
		(603)	実行サーバ1	実行サーバ2	実行サーバ3	…
	実行サーバ1	CPU	5	-	-	
		メモリ	5	-	-	
		ディスク	5	15	-	
		データベース	10	20	-	
		…				
	実行サーバ2	CPU	-	5	-	
		メモリ	-	5	-	
		ディスク	-	5	15	
		…				
	実行サーバ3	CPU	-	-	10	
		メモリ	-	-	5	
		ディスク	15	20	10	
		データベース	15	30	10	
		…				
	リソース1	ディスク	10	-	40	
	リソース2	ディスク		20	40	
	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

(604)　(605)

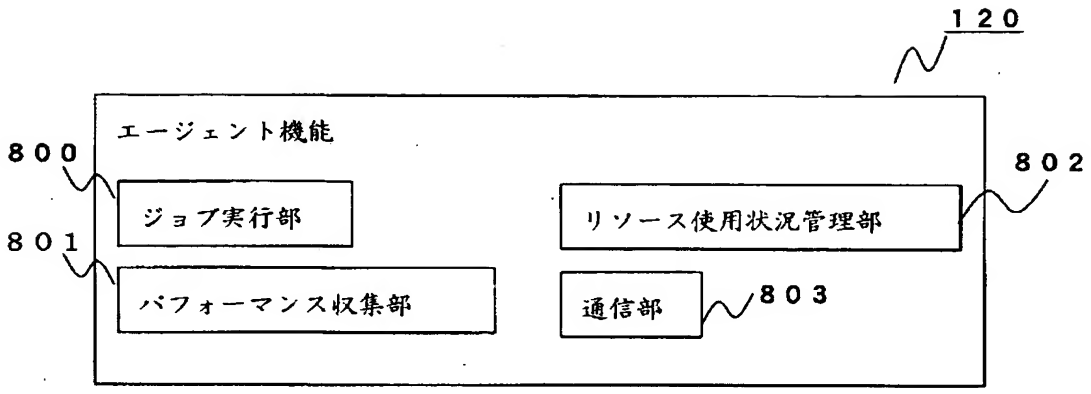
【図７】

図７


開始
700
NO
エージェント新規登録依頼あり
701
YES
新規エージェント登録
NO
パフォーマンス収集時間
702
YES
703
パフォーマンス情報送信要求
704
リソース間距離テーブル更新
705
NO
ジョブ実行時間
YES
706
ジョブ実行
終了

【図８】

図８



【図９】

図９

履歴パフォーマンステーブル　９００

９０１ 収集時刻	９０３ 収集項目 CPU %	使用メモリ量	空きディスク容量	９０２ 平均ディスク処理時間	９０４ 平均クエリ処理時間	９０５ …
10:00:01	30%	200MB	1000MB	0.5ms	0.01ms	
10:00:02	23%	256MB	900MB	0.2ms	0.02ms	
10:00:03	10%	128MB	900MB	0.6ms	0.05ms	
…						
11:00:00	40%	200MB	500MB	1.2ms	0.25ms	
11:00:01	14%	250MB	700MB	0.3ms	0.10ms	
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

【図１０】

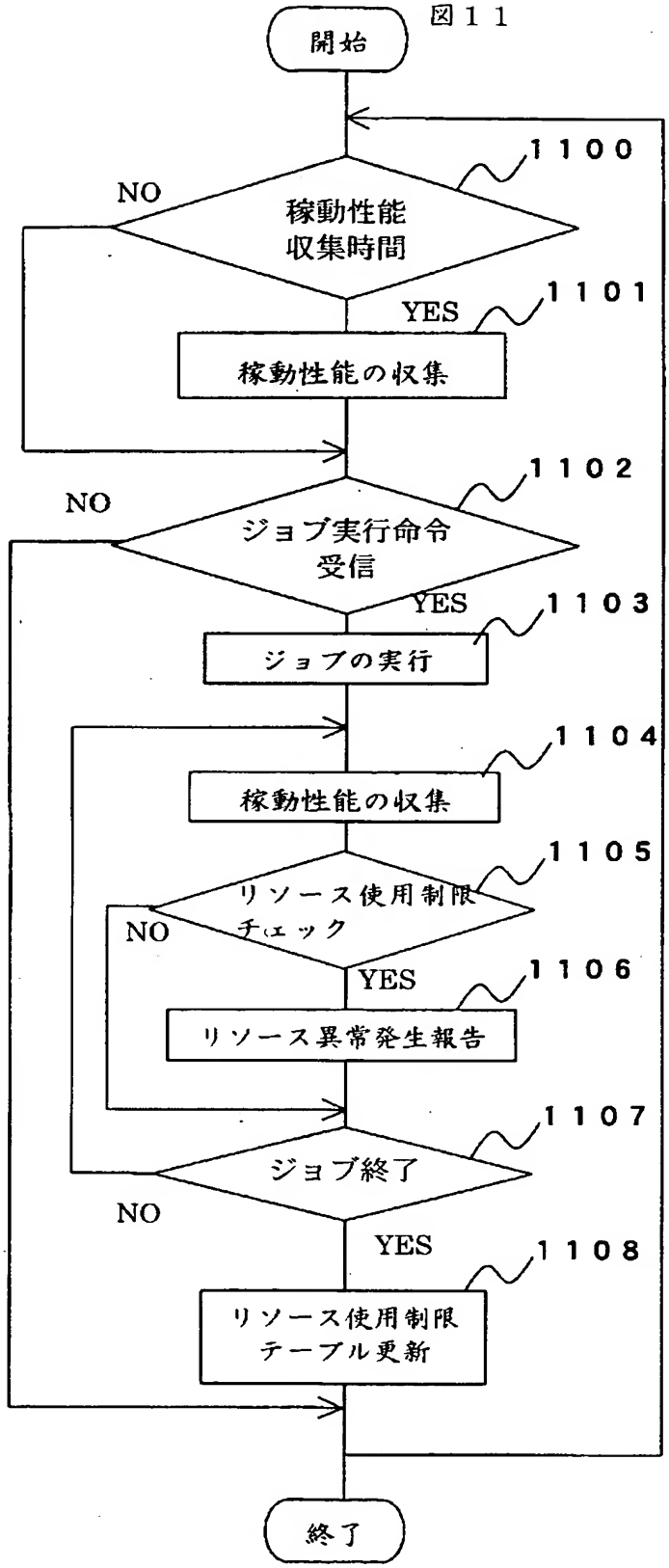
図１０

リソース使用制限テーブル　１０００

ジョブ (1001)	しきい値越え回数 (1003)	しきい値 (1002)			
		CPU使用率	使用メモリ量	空きディスク容量	…
ジョブ1	2/3回	50%以上	384MB以上	200MB以下	
ジョブ2	2/3回	60%以上	-	300MB	
ジョブ3	1/1回	80%以上	-	-	
	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

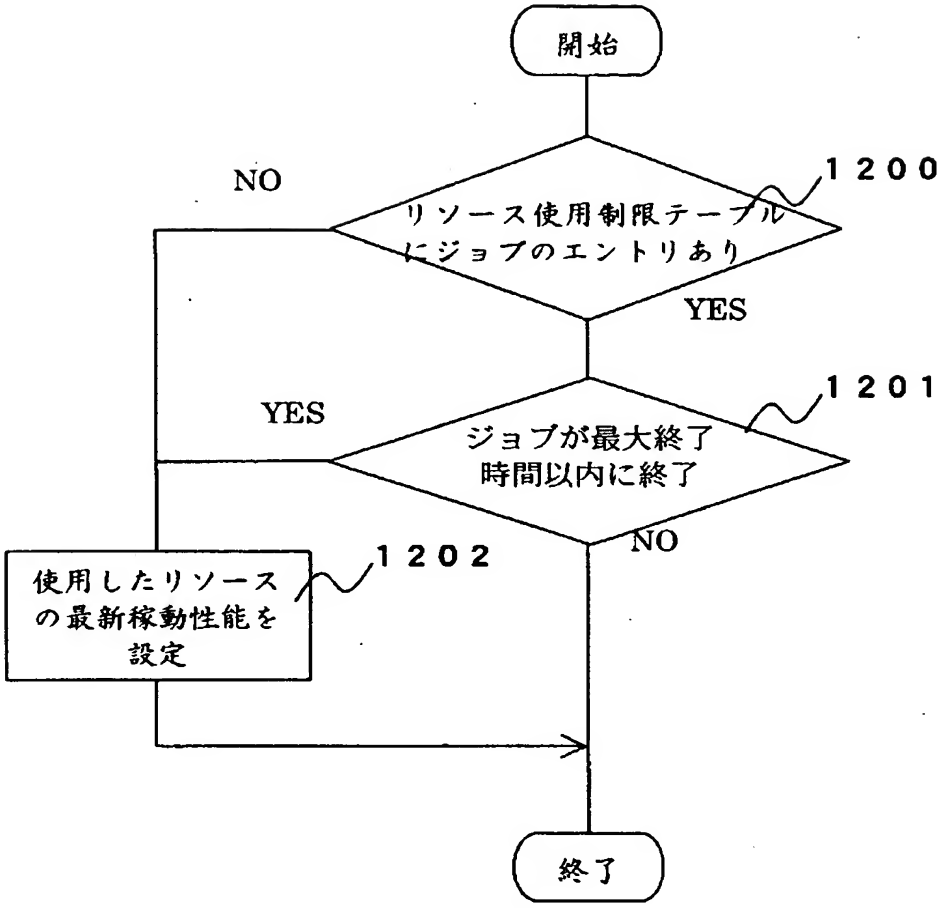
【図１１】

図１１

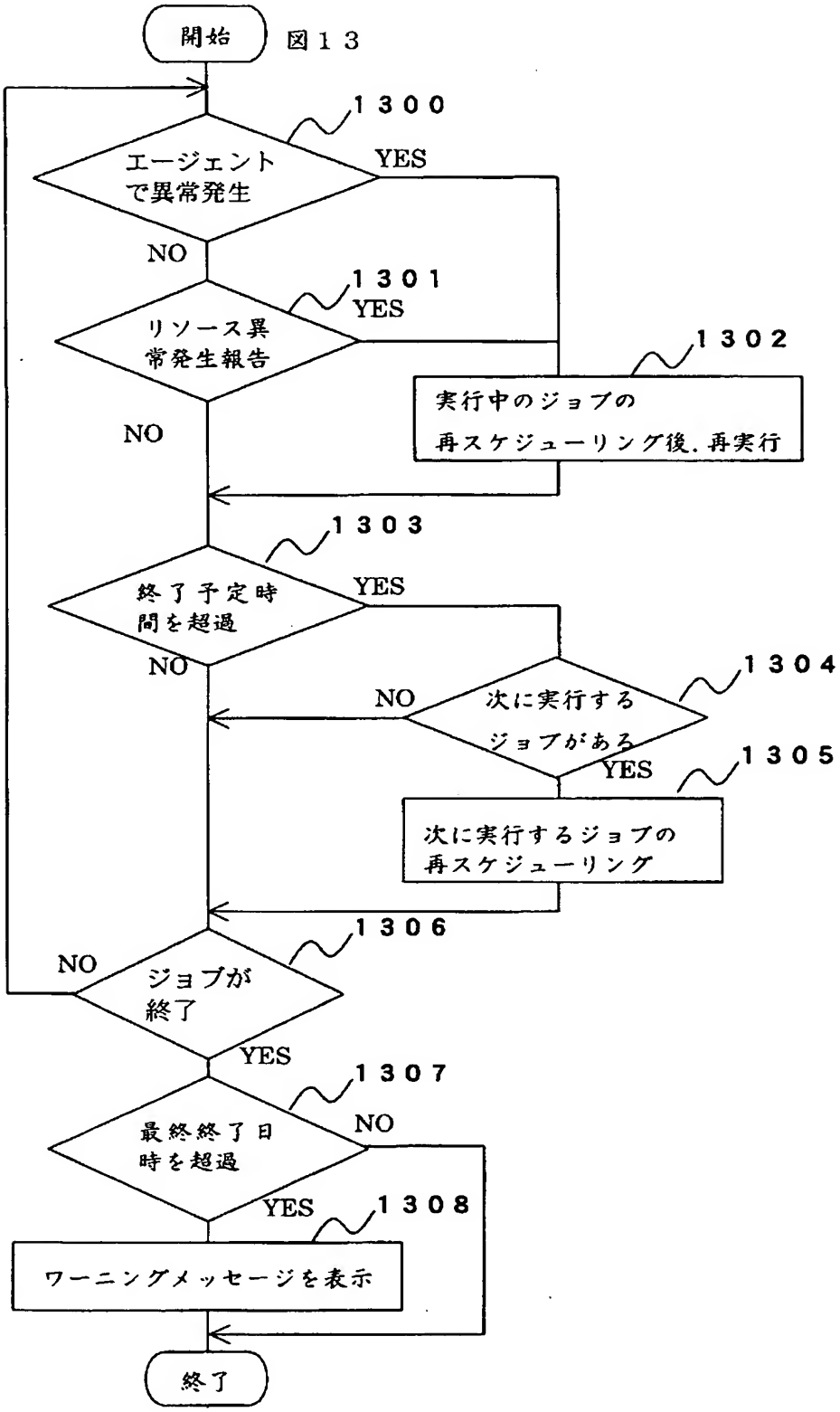

開始
1100
稼動性能
収集時間
NO
YES
1101
稼動性能の収集
1102
ジョブ実行命令
受信
NO
YES
1103
ジョブの実行
1104
稼動性能の収集
1105
リソース使用制限
チェック
NO
YES
1106
リソース異常発生報告
1107
ジョブ終了
NO
YES
1108
リソース使用制限
テーブル更新
終了

【図１２】

図１２

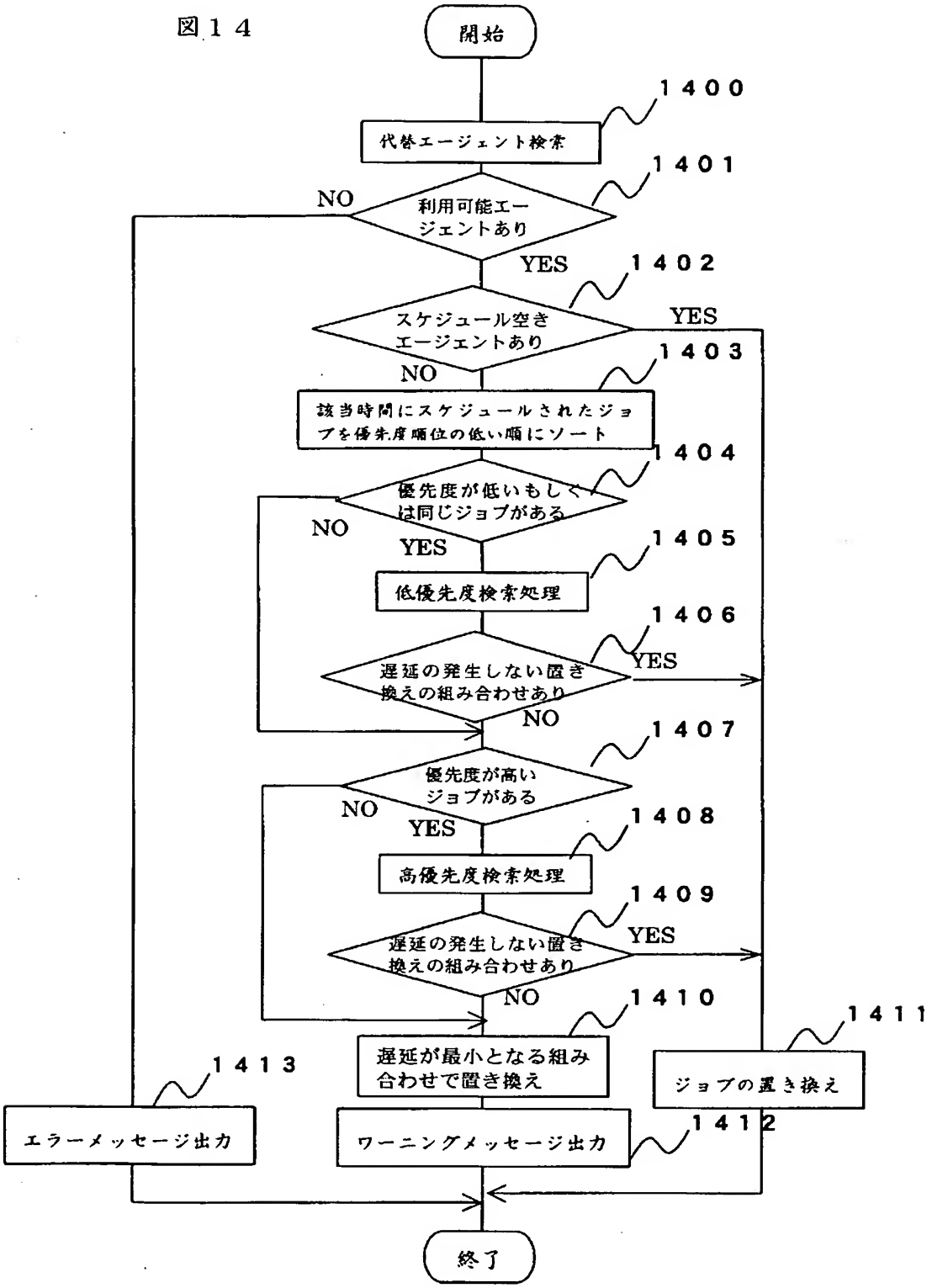

開始
1200
リソース使用制限テーブル
にジョブのエントリあり
NO
YES
1201
ジョブが最大終了
時間以内に終了
YES
NO
1202
使用したリソース
の最新稼動性能を
設定
終了

【図１３】


開始
図１３
１３００
エージェントで異常発生
YES
NO
１３０１
リソース異常発生報告
YES
１３０２
実行中のジョブの再スケジューリング後、再実行
NO
１３０３
終了予定時間を超過
YES
NO
１３０４
次に実行するジョブがある
NO
YES
１３０５
次に実行するジョブの再スケジューリング
１３０６
ジョブが終了
NO
YES
１３０７
最終終了日時を超過
NO
YES
１３０８
ワーニングメッセージを表示
終了

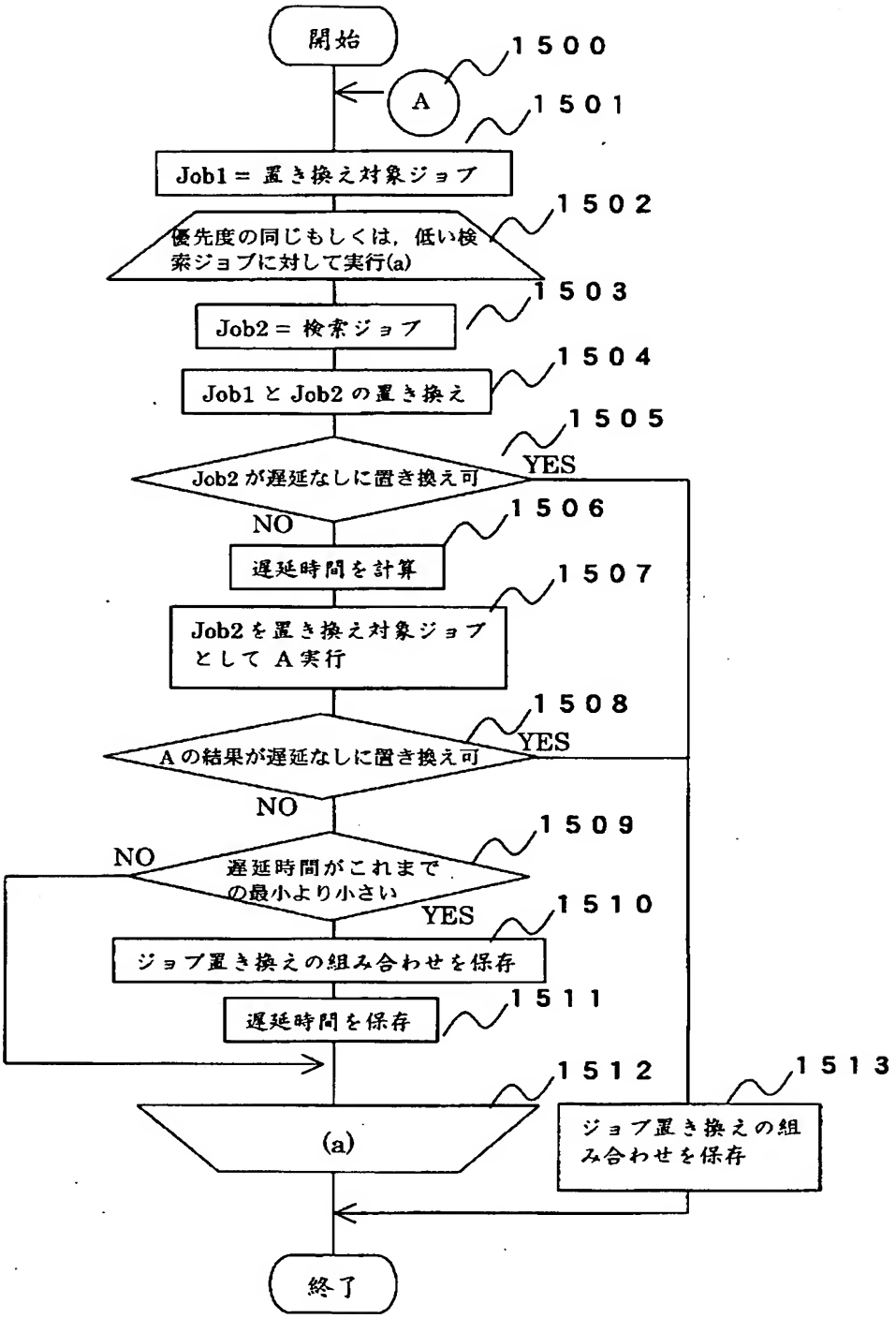
【図１４】

図１４


開始
1400
代替エージェント検索
1401
利用可能エージェントあり
NO
YES
1402
スケジュール空きエージェントあり
YES
1403
NO
該当時間にスケジュールされたジョブを優先度順位の低い順にソート
1404
優先度が低いもしくは同じジョブがある
NO
YES
1405
低優先度検索処理
1406
遅延の発生しない置き換えの組み合わせあり
YES
NO
1407
優先度が高いジョブがある
NO
YES
1408
高優先度検索処理
1409
遅延の発生しない置き換えの組み合わせあり
YES
NO
1410
遅延が最小となる組み合わせで置き換え
1411
ジョブの置き換え
1413
エラーメッセージ出力
ワーニングメッセージ出力
1412
終了

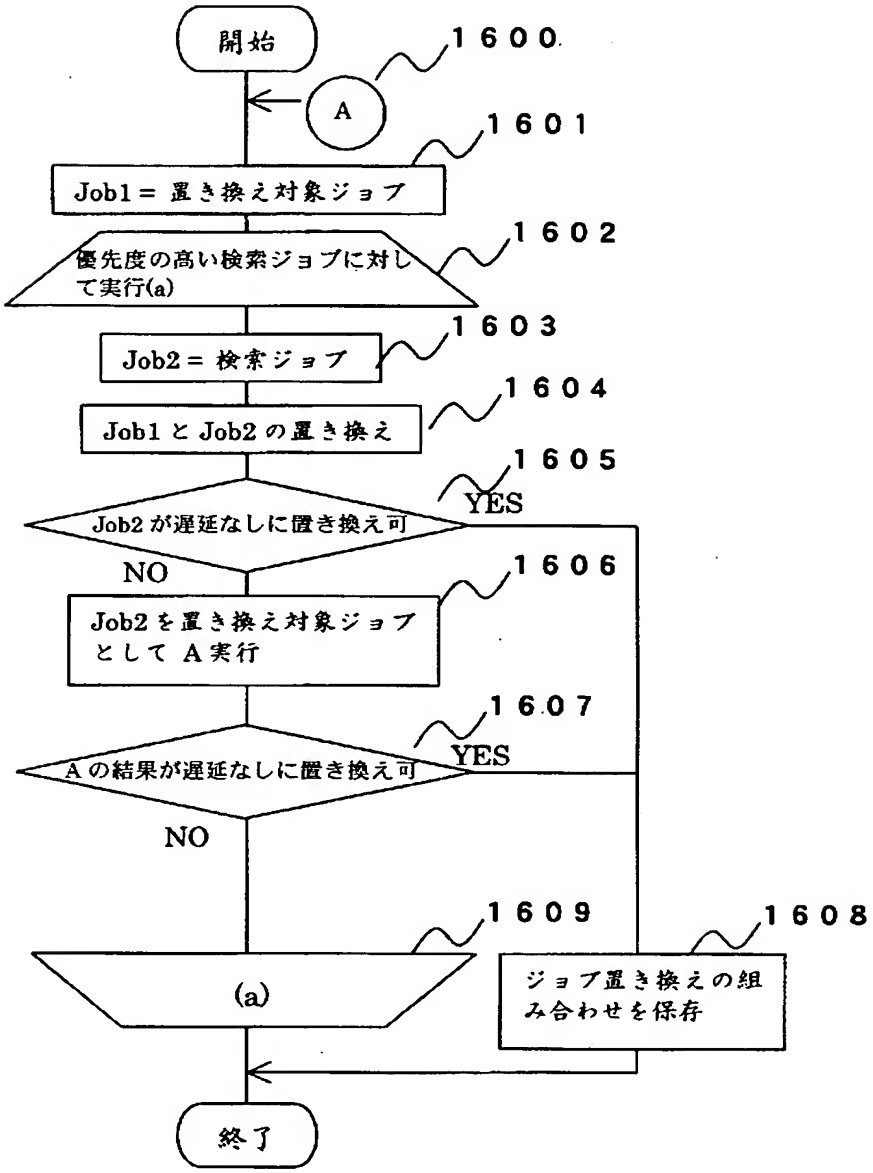
【図１５】

図１５


開始
1500
A
1501
Job1＝置き換え対象ジョブ
1502
優先度の同じもしくは，低い検索ジョブに対して実行(a)
1503
Job2＝検索ジョブ
1504
Job1とJob2の置き換え
1505
Job2が遅延なしに置き換え可
YES
NO
1506
遅延時間を計算
1507
Job2を置き換え対象ジョブとしてA実行
1508
Aの結果が遅延なしに置き換え可
YES
NO
1509
遅延時間がこれまでの最小より小さい
NO
YES
1510
ジョブ置き換えの組み合わせを保存
1511
遅延時間を保存
1512
(a)
1513
ジョブ置き換えの組み合わせを保存
終了

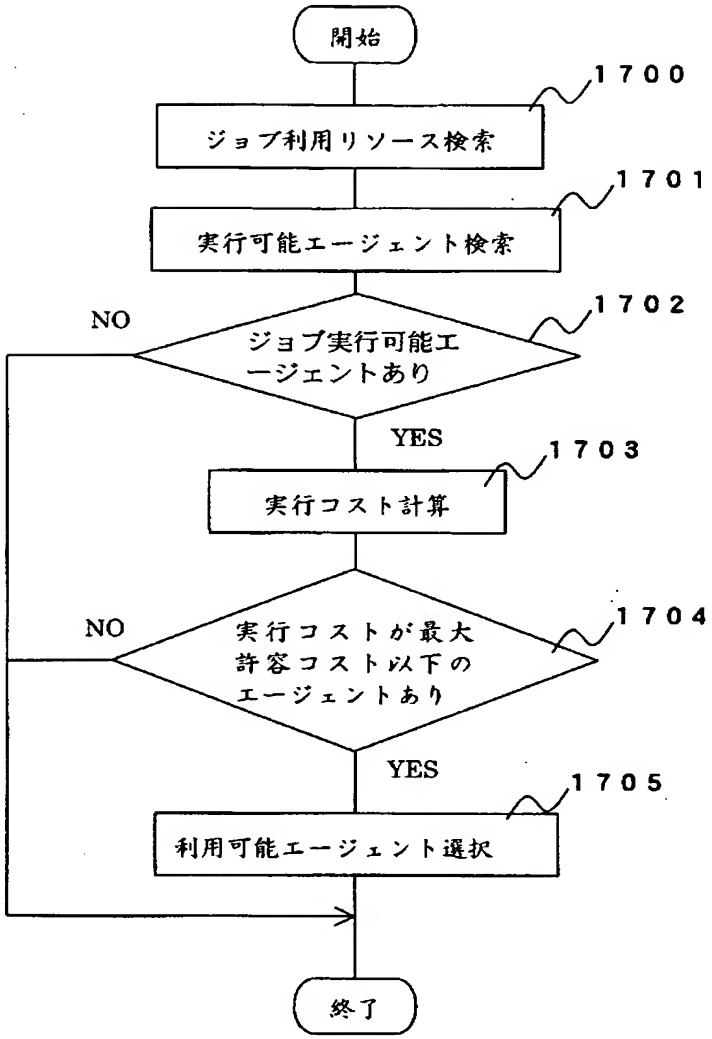
【図１６】

図１６


開始
1600
A
1601
Job1＝置き換え対象ジョブ
1602
優先度の高い検索ジョブに対して実行(a)
1603
Job2＝検索ジョブ
1604
Job1とJob2の置き換え
1605
YES
Job2が遅延なしに置き換え可
NO
1606
Job2を置き換え対象ジョブとしてA実行
1607
YES
Aの結果が遅延なしに置き換え可
NO
1609
1608
(a)
ジョブ置き換えの組み合わせを保存
終了

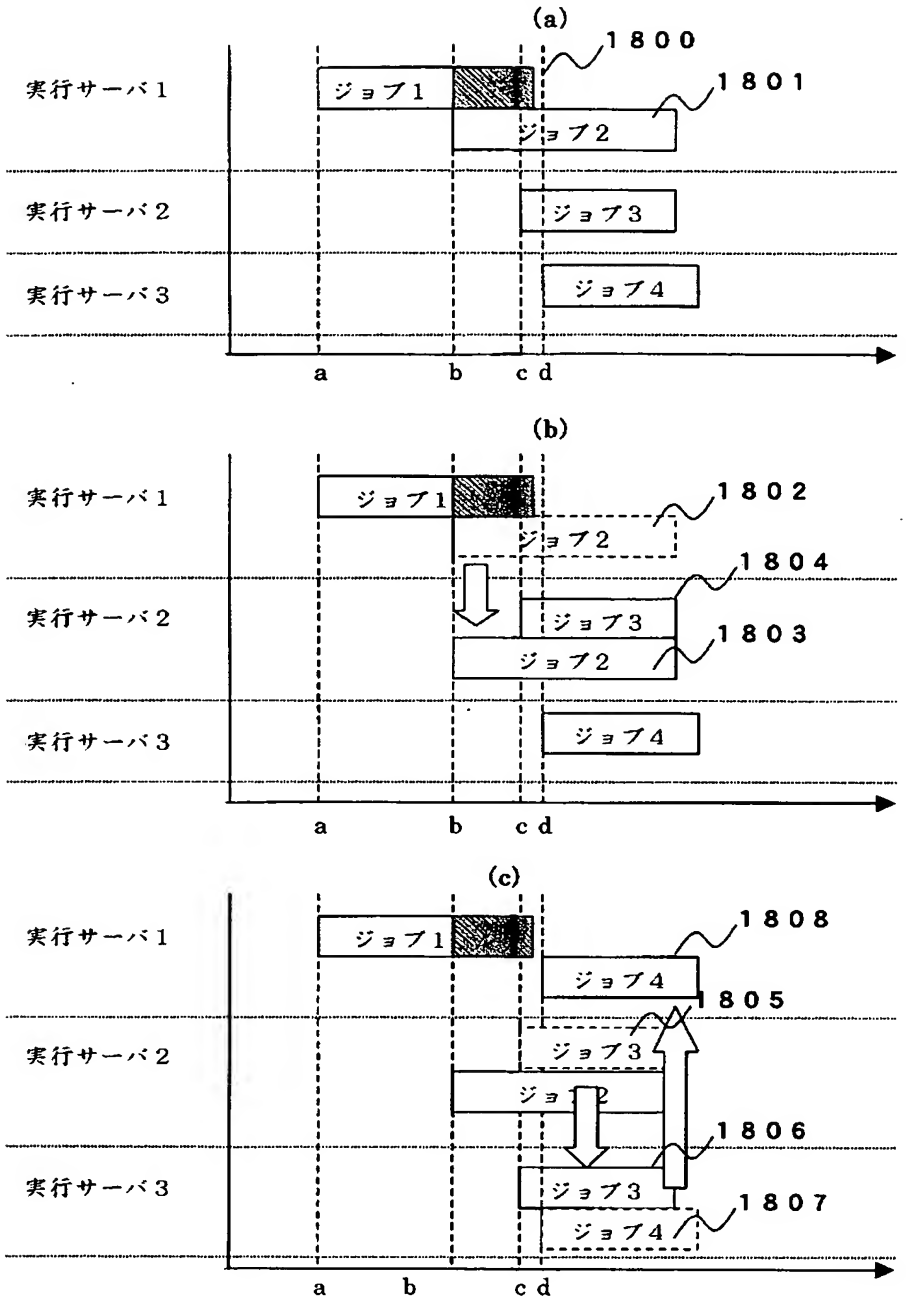
【図１７】

図１７


開始
1700
ジョブ利用リソース検索
1701
実行可能エージェント検索
1702
NO
ジョブ実行可能エージェントあり
YES
1703
実行コスト計算
1704
NO
実行コストが最大許容コスト以下のエージェントあり
YES
1705
利用可能エージェント選択
終了

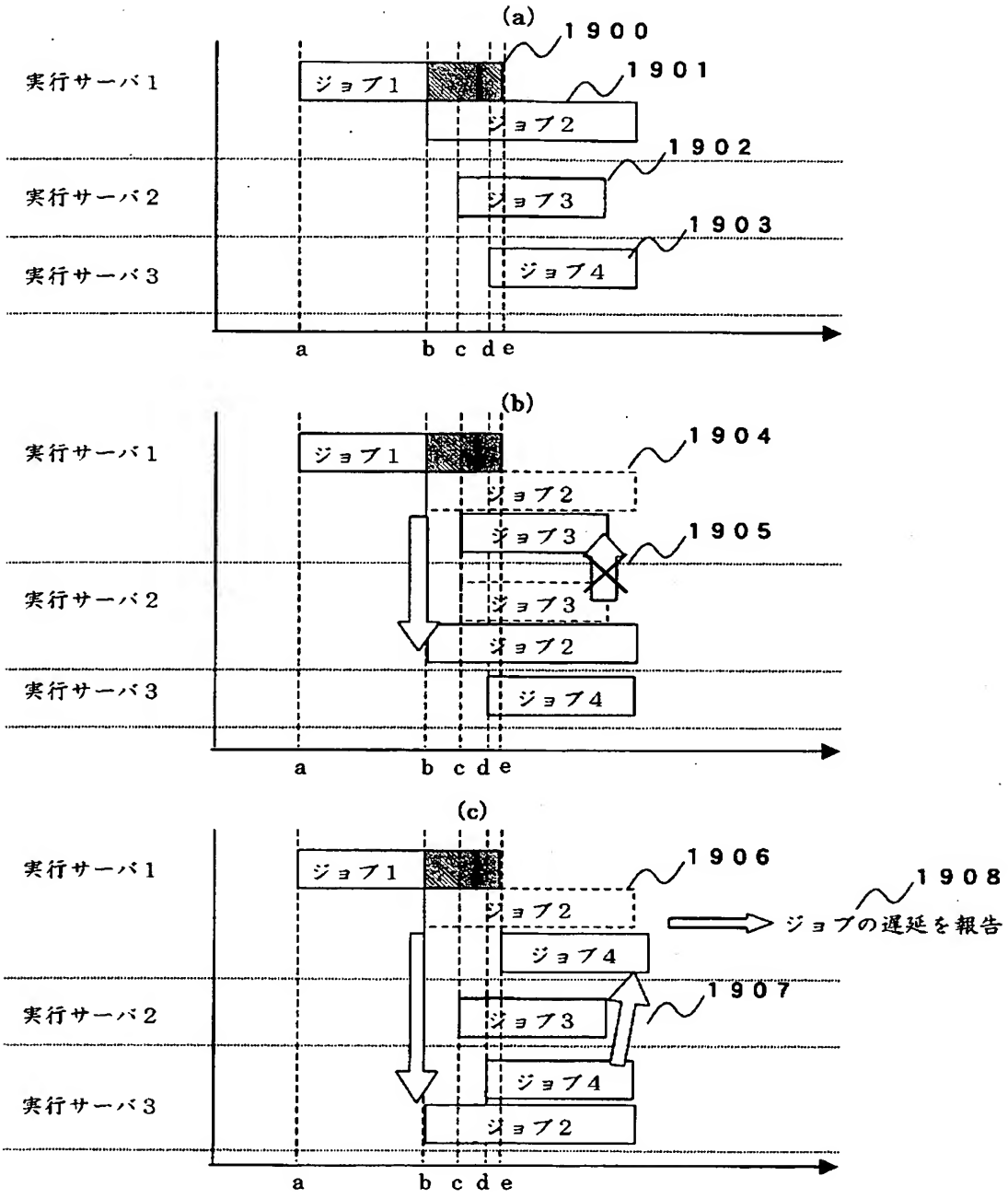
【図１８】

図１８


(a)
1800
1801
実行サーバ1
ジョブ1
ジョブ2
実行サーバ2
ジョブ3
実行サーバ3
ジョブ4
a
b
c d
(b)
1802
1804
1803
実行サーバ1
ジョブ1
ジョブ2
実行サーバ2
ジョブ3
ジョブ2
実行サーバ3
ジョブ4
a
b
c d
(c)
1808
1805
1806
1807
実行サーバ1
ジョブ1
ジョブ4
実行サーバ2
ジョブ3
ジョブ2
実行サーバ3
ジョブ3
ジョブ4
a
b
c d

【図１９】

図１９


(a)
1900
1901
1902
1903
実行サーバ１
実行サーバ２
実行サーバ３
ジョブ１
ジョブ２
ジョブ３
ジョブ４
a b c d e
(b)
1904
1905
実行サーバ１
実行サーバ２
実行サーバ３
ジョブ１
ジョブ２
ジョブ３
ジョブ３
ジョブ２
ジョブ４
a b c d e
(c)
1906
1908
ジョブの遅延を報告
1907
実行サーバ１
実行サーバ２
実行サーバ３
ジョブ１
ジョブ２
ジョブ４
ジョブ３
ジョブ４
ジョブ２
a b c d e

【書類名】　要約書

【要約】

【課題】

ジョブを実行する実行サーバで障害や性能の劣化が発生した場合にジョブのスケジュールを再構成する技術を提供する。

【解決手段】

ジョブのスケジュールを管理するジョブスケジュール管理方法であって、ジョブが割り当てられた計算機の稼動状況を監視し、稼動状況が所定の条件を満たすか否か判断し、稼動状況が所定の条件を満たす場合、計算機に割り当てられたジョブのうち、所定の条件を満たした時点で未完了のジョブを検出し、検出した未完了のジョブの実行に必要なリソース情報に基づいて、検出した未完了のジョブを実行可能な他の計算機を抽出し、検出した未完了のジョブを抽出した他の計算機に割り当てる。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００３－１９３２５９
受付番号	５０３０１１３０１２５
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　７月　９日

＜認定情報・付加情報＞

【提出日】　平成15年 7月 8日

特願２００３－１９３２５９

出　願　人　履　歴　情　報

識別番号　　［０００００５１０８］

１．変更年月日　　１９９０年　８月３１日
［変更理由］　　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所